Python tutorial - Looking for Mergers using Statmorph

April 4, 2023

Ananya P. Sreelekha

Advisor: Dr. Katherine E Whitaker

1 Abstract

The purpose of this Python Tutorial is to intoduce `statmorph` to help perform statistical analysis on galaxy morphology. In particular, we will be demonstrating how to extract the *Gini* and M_{20} statistics and use it to make a *Gini* (x-axis) vs. M_{20} (y-axis) plot (Lotz et al. 2004) to identify galaxy merger population. This is done by implementing a simple `for` loop to iterate over all the selected items in the catalog. Major mergers can result in a period of rapid star formation (starburst), followed by a shut down in star formation rates making the galaxies quiescent. In this tutorial we make a selection of galaxies by first creating a U-V vs V-J color plot to pick out quiescent galaxies (Whitaker et. al 2012), and next use *Gini* - M_{20} mergers to determine if they are mergers. Mergers comprise one viable pathway to quiescence.

2 Introduction

The James Webb Space Telescope (JWST) is 100 times more powerful than the Hubble Space Telescope (HST), capturing higher resolution and deeper images of galaxy morphologies. Morphological analyses are therefore more reliable and large samples enable us to implement statistical analyses to quantify morphology. This tutorial uses an in-house photometric catalog of the Cosmic Early Evolution Release Survey (CEERS) which is an extragalactic survey that covers 100sq arcmins of the Extended Groth Stip (EGS) field with JWST imaging and spectroscopy in NIRCam, NIRSpec and MIRI. Here we use `statmorph`, an affiliated package of `astropy` to perform non-parametric morphological diagnostics on galaxy images.

The morphology of a galaxy can give us a lot of information about its evolution history and composition. Mergers are one way by which galaxies evolve. The effects of a merger are evident in a galaxy's morphology that can be seen as tidal tails formed by gravitational disturbances, asymmetry in shape, a range of radial light profiles etc. Gini coefficient is a statistic used in economics to describe the wealth distribution among a population. We use the same statistic in astronomy to describe the distribution of flux throughout the galaxy. M_{20} statistic (Lotz et al. 2004) measures the second moment of a galaxy's brightest regions, containing 20% of the total flux, relative to the total second-order central moment. Plotting *Gini* coefficient vs M_{20} (eg. Fig. X in Lotz et al.2004) for the galaxies can help us easily distinguish mergers from non mergers.

Ellison et al. 2022 show that galaxy mergers can lead to a shut down in star formation (quiescence) after a brief period of rapid star formation (starburst). Quiescent galaxies can be distinguished

from star forming galaxies by plotting a U-V vs. V-J diagram. The U-V color is the difference in magnitude of restframe ultra-violet (U) (0.36 microns approx) and visual (V) (0.55 microns approx) bands and V-J is the difference in magnitude of restframe visual and infrared (J)(1.2 microns approx.) bands. While most star forming galaxies are bluer, sometimes the star forming regions can be dust obscured making it appear redder. Galaxies can also be reddened by old age and quiescence. A UVJ plot is a robust way of distinguishing red quiescent galaxies from dusty star forming galaxies (Fang,J et al.2018). Mergers can usually explain the presence of large quiescent galaxies at the redshift of $1 < z < 2$ (Bruce et al. 2012). Therefore, after applying our initial selection of quiescent galaxies make an additional selection for galaxies at a redshift range of $0.5 < z < 2.5$ and with a solar mass of 10^{10} or greater.

This tutorial starts by making a selection of galaxies based on Whitaker et al. 2012 after plotting a U-V vs. V-J diagram for all objects in the CEERS catalog that is at a redshift(z) of $0.5 < z < 2$ and a mass of 10^{10}. After noting their object IDs in the catalog from the U-V vs. V-J plot, we use the ID list of quiescent galaxies to plug into `statmorph`, which will generate *Gini* and M_{20} values to make a plot.

3 Data

The data used in this tutorial can be found under the google drive link to our internal CEERS catalog: https://drive.google.com/drive/folders/1T5YZiy1UJOkBgRCIREmAqhSyd7ZkfF4k?usp=share_link

Filenames:

Science image : ceers-full-grizli-v4.0-f444w-clear_drc_sci_skysubvar.fits [805.3MB]

Photometric Catalog : LW_f277w-f356w-f444w_SCIREADY_CATALOG.fits [9.3MB]

Redshift Catalog : ceers_LW_f160w_v4.zout.fits [17.3MB]

Segmentation map : LW_f277w-f356w-f444w_SEGMAP.fits [805.3MB]

Weight map : ceers-full-grizli-v4.0-f444w-clear_drc_wht.fits [314.5MB]

Point spread function : psf_ceers_F444W_4arcsec.fits [86KB]

We can use other extragalactic catalogs, just make sure they have the standard information on the coordinates, mass and photometric data (rest frame U, V and J magnitudes, redshift). Here is a link to other 3D HST catalogs. Information about photometric data can usually be found in the header of the file. Header information can be viewed by opening the file using a text editor (not for `.fits` files). For `.fits` files, the header information can be accessed by following the steps listed in step 2 under the Method section.

Note: If you need to utilize more than one catalog file to get all the information needed for this tutorial, make sure to check that the number of objects in the catalog and object id match

4 Method

1. Before we start using the data from the catalog, import necessary utilities from `astropy`, `statmorph` and `photutils` for data visualization, analysis and photometry. General modules such as `numpy` to store/create values into arrays for easier data manipulation, `pandas` for handling dataframes and `matplotlib.pyplot` for plotting also need to be imported. A

short description of why these `statmorph` and `astropy` packages are useful and link to the documentation are listed below:

- `source_morphology` from `statmorph` : function that runs morphological diagnostics on galaxy images

- `astropy.units` : for handling units and unit conversions
- `astropy.wcs` : required for World Coordinate System (WCS) transformation between one set of coordinates to another.
- `astropy.table` : provides functionality for storing and manipulating heterogenous tables
- `simple_norm` from `astropy.visualization` : normalization class used to display images on Matplotlib
- `models` from `astropy.modeling` : for making models from a class of previously defined models
- `convolve` from `astropy.convolution` : convolve arrays with kernel (with interpolation for wherever there are NaN values)
- `Cutout2D` from `astropy.ndata` : used for creating cutouts of an image file
- `fits` from `astropy.io` : helps handle fits files

[1]:
```
# general modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import photutils
import scipy.ndimage as ndi #for multidimensional image processing

# statmorph
import statmorph
from statmorph import source_morphology

#astropy packages
import astropy.units as u
from astropy.wcs import WCS
from astropy.coordinates import SkyCoord
from astropy.table import Table, Column , join
from astropy.visualization import simple_norm
from astropy.modeling import models
from astropy.convolution import convolve
from astropy.nddata import Cutout2D
from astropy.io import fits

# package to suppress warnings in functions (use if needed)
import warnings

%matplotlib inline
```

2. Load in data files, including catalog, segmentation map, weight map and point spread function (psf) using `fits.getdata(filepath/filename)`. An example for filepath could be

`/Users/Guest/Documents/Lab`. If the files are saved in a different folder than the Jupyter Notebook, you WILL NEED TO give the file path for each of the data files. For example If it is one directory above we can use `../filename`. We need one more file that contains Header Data Unit List (HDUL) for the catalog fits file. The HDUL usually has information about headers and contents can be viewed by running `hdul.info()`.

Note: For catalog files, we need to use `Table.read(filepath/filename)` *even though these files may may be saved as a* `.fits` *extension. An alternative way is to use* `ascii.read(filepath/filename)`. *In the code below I only use filenames as this Jupyter Notebook was saved in the same directory as the files*

[2]:
```
hdul = fits.open('ceers-full-grizli-v4.0-f444w-clear_drc_sci_skysubvar.fits')
ceers = Table.read('LW_f277w-f356w-f444w_SCIREADY_CATALOG.fits')
ceers_z = Table.read('ceers_LW_f160w_v4.zout.fits')
seg = fits.getdata('LW_f277w-f356w-f444w_SEGMAP.fits')
wht_map = fits.getdata('ceers-full-grizli-v4.0-f444w-clear_drc_wht.fits')
psf = fits.getdata('psf_ceers_F444W_4arcsec.fits')

sci = hdul[1].data #science image
w = WCS(hdul[1].header) #header information
```

3. After data files are read/loaded you can try printing the first few rows to examine its contents and column names. We can print just the column names to check if the catalog contains the values we need for analysis. If some values are split between 2 files, we need to make sure the length and object IDs match for the files. This can be done with `len(filename)` and then printing out just the column with the object ids in both files.

Note: Although this step may seem trivial, it helps to get a good idea of how the catalog is structured. For example, printing the column names will be useful in the next steps when we slice the data based on the quantities in each column

[3]:
```
# print catalog to check
print(ceers_z[0:5]) #catalog with redshift values
print(ceers.columns) #print only column header
```

```
 id         ra         …                  ABSM_274
           deg         …
--- ----------------- … ----------------------------------------
  1 214.92376581009228 …                                inf .. inf
  2 214.91865978302255 …                  -- .. -4.419780331242023
  3 214.91774449179488 … -15.903354580345862 .. -17.474210674706214
  4  214.9171840298942 … -15.694146914049881 .. -18.178157610461014
  5 214.91703042102276 …                                   -- .. --
<TableColumns names=('id','x','y','ra','dec','ebv_mw','faper_f444w','eaper_f444w
','f_f435w','e_f435w','f_f606w','e_f606w','f_f814w','e_f814w','f_f105w','e_f105w
','f_f125w','e_f125w','f_f140w','e_f140w','f_f160w','e_f160w','f_f115w','e_f115w
','f_f150w','e_f150w','f_f200w','e_f200w','f_f277w','e_f277w','f_f410m','e_f410m
','f_f356w','e_f356w','f_f444w','e_f444w','tot_ekron_F444w','tot_cor','z_spec','
star_flag','kron_radius','a_image','b_image','theta_J2000','flux_radius','use_ph
```

```
ot')>
```

[4]:
```
#Get length of files
print(len(ceers))
print(len(ceers_z))
```

```
25172
25172
```

[5]:
```
#print column with object id
print(ceers['id'][0:3])
print(ceers_z['id'][0:3])
```

```
 id
---
  1
  2
  3
 id
---
  1
  2
  3
```

4. Define selection for redshift range and mass. We will use redshift of $0.5 < z < 2.5$ and a mass of 10^{10} solar masses or greater. These values are reasonable because massive ($>10^{10}$ solar masses) quiescent galaxies around $1 < z < 2$ are explained by mergers (Whitaker et al.2012) and we want to select galaxies around this range.

Note: The selection also includes `ceers[use_phot==1]` which will filter out all the unreliable objects.

[6]:
```
# selection based on redshift and mass, include ceers[use_phot==1] to avoid␣
 ↪unreliable data
selection = (ceers['use_phot'] == 1)&(ceers_z['z_phot']>0.
 ↪5)&(ceers_z['z_phot']<=2.5)&(ceers_z['mass']>=1e10)
```

5. Compute U-V and V-J values by using restframe U, V and J data from catalog. These are given in columns `['restU']`,`['restV']` and `['restJ']` columns respectively. We will first convert the values to magnitudes and find U-V and V-J color. The formula for difference in magnitude is given by:

$m_1 - m_2 = -2.5 log_{10} \frac{m_1}{m_2}$

Note: There may be divide by zero warnings, it is possible to supress them by writing `warnings.filterwarnings("ignore")`

[7]:
```
warnings.filterwarnings("ignore")
U_V = -2.5*np.log10(ceers_z['restU']/ceers_z['restV'])
V_J = -2.5*np.log10(ceers_z['restV']/ceers_z['restJ'])
```

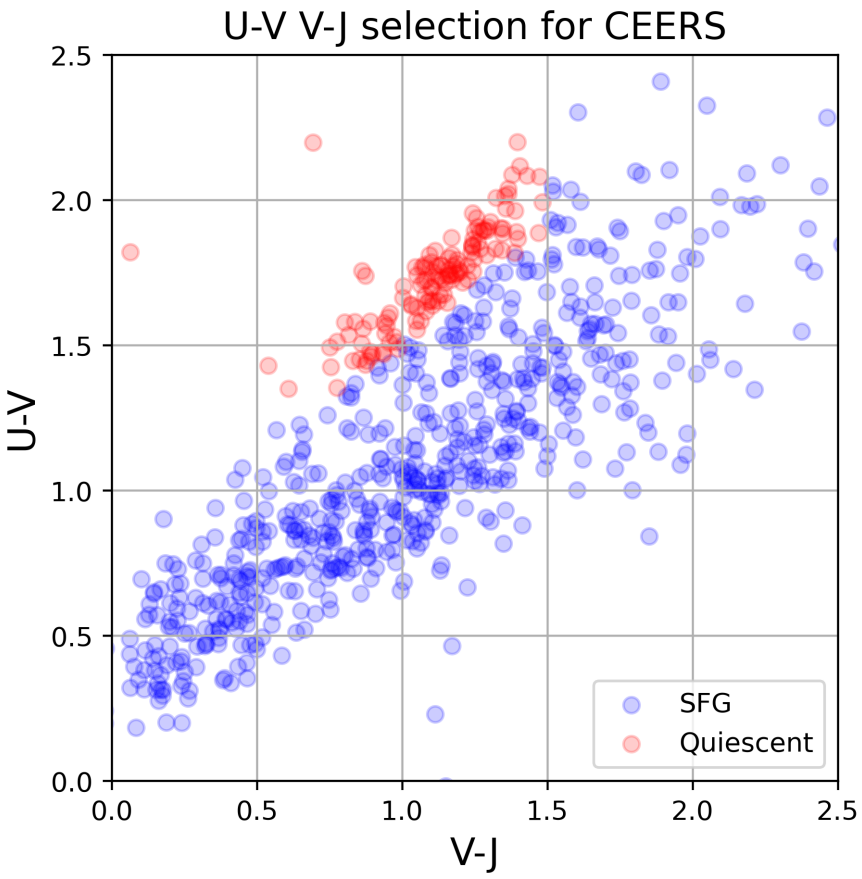
5. Define selection for quiscent galaxy as given in Whitaker et al.2012. The function takes in U-V and V-J (arrays) and classifies them as quiescent (1) or star forming (0). After applying the selection function, we will have a sample of galaxies to run on `statmorph`, and we can quickly visualize it using a scatter plot of U-V (y-axis) vs. V-J (x-axis).

[8]:
```
# function to select quiescent galaxies
def sel_quiescent(u_v,v_j):
    out = np.zeros(len(u_v))
    crit_w12 = (u_v > (0.8 * v_j + 0.7)) & (u_v > 1.3) & (v_j < 1.5)
    out[crit_w12] = 1
    return out
```

[9]:
```
# use quiescent function defined in the previous cell
sel_q = sel_quiescent(U_V,V_J)
```

[10]:
```
# plot to visualize quiescent selection
plt.figure(figsize=(5,5),dpi=400,facecolor='white')

plt.scatter(V_J[selection&(sel_q==0)],U_V[selection&(sel_q==0)],alpha=0.2,␣
 ↪color='b',label='SFG')
plt.scatter(V_J[selection&(sel_q==1)],U_V[selection&(sel_q==1)],alpha=0.2,␣
 ↪color='r',label='Quiescent')
plt.xlabel('V-J',fontsize=14)
plt.ylabel('U-V',fontsize=14)
plt.title('U-V V-J selection for CEERS',fontsize=14)
plt.axis('square')
plt.xlim([0,2.5])
plt.ylim([0,2.5])
plt.grid()
plt.legend()
```

[10]:
```
<matplotlib.legend.Legend at 0x7f78d1346af0>
```



6. Grab object IDs of the sources from the catalog that are identified as quiescent.

[11]:
```
# list of object IDs of quiescent galaxies
objid = ceers['id'][selection&(sel_q==1)]
```

7. Run the objects selected above on `source_morphology` function by `statmorph` by iterating through all the objects using a `for` loop. Make empty lists for the storing the *Gini* and M_{20} values returned by the `source_morphology` function. The function also returns the flags for sersic parameters, and `flag_sersic==1` indicates sersic parameters computed are less reliable. This value is usualy 1 in the case of mergers because the function breaks when there is an irregular morphology. So storing the `flag_sersic` will increase will help check reliability.

[12]:
```
gini = []
m_20 = []
flag_sersic = []
```

8. We create a `for` loop that will go over each object ID one by one. The `source_morphology` function also takes in the segmentation map, weight maps and psf as arguments. A segmentation map has values for every pixel equal to the ID that indicate if pixel is a detected source

or a background pixel. The weight map gives the 1/variance ($\frac{1}{\sigma^2}$) value for each pixel. The psf is used to correct images for the spreading of light when light enters the telescope and hits the detector. Inside the loop, we make a cutout for each object with a box size of 5 $arcsecs^2$. The cutout size and region should be matched for the weight map, segmentaion map and the psf. Due to the cutout size and region, the segmentation map may end up throwing an error so to avoid this, we set all pixels to 0 (indicates background). The *Gini* coefficient, M_{20} and `flag_sersic` are stored in a `pandas` dataframe. The `try` and `except` conditions have been implemented to resolve the non-finite value error that is produced due to issues with the segmentation map.

Note: I highly recommend using the ignore warnings code commented out in the beginning of the cell to avoid being spammed by failed sersic fits. You are also free to create lists of other statistics that the code returns. This happens very often (almost every other fit) due to the fact that sersic fits are often unreliable in the case of merger candidates

[13]:
```
#warnings.filterwarnings("ignore")
i = 0
for ob in objid:
    print(i)
    i+=1
    obj = ceers[ceers['id']==ob]
    pos = SkyCoord(obj['ra'],obj['dec'])
    image = Cutout2D(sci,pos,5*u.arcsec,wcs=w).data
    objseg = Cutout2D(seg.copy(),pos,5*u.arcsec,wcs=w).data
    wht = Cutout2D(wht_map.copy(),pos,5*u.arcsec,wcs=w).data
    # some objects tend to return a non-finite value error due to the␣
↪segmentation map. So, we will try running
    #source morphology, and if there is an error, skip to the next if there is␣
↪a non-finite value error
    try:
        source_morphs = statmorph.source_morphology(image, objseg, label=ob,␣
↪weightmap=wht, psf=psf)
        morph = source_morphs[0]
        gini.append(morph.gini)
        m_20.append(morph.m20)
        flag_sersic.append(morph.flag_sersic)
    except:
        print('Cannot do it!')
        #set a constant error value for Gini and M20 whenever there fit cannot␣
↪be done, and
        #set flag to 1 to indicate bad fit
        gini.append(-99)
        m_20.append(-99)
        flag_sersic.append(1)
        continue
```

```
0
1
```

```
2
Cannot do it!
3
Cannot do it!
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Cannot do it!
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
Cannot do it!
36
Cannot do it!
37
38
39
40
41
42
43
44
```

```
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
Cannot do it!
65
66
67
68
Cannot do it!
69
70
71
72
Cannot do it!
73
74
75
76
77
78
79
80
Cannot do it!
81
82
83
84
85
86
87
88
```

```
89
90
91
92
93
94
95
96
97
98
Cannot do it!
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
Cannot do it!
115
116
117
118
Cannot do it!
119
120
121
122
123
124
125
126
127
128
129
130
Cannot do it!
131
```

[14]:
```
# print table created
table = pd.DataFrame({'object_id':objid, 'gini':gini, 'M_20':m_20,'flag_sersic':
 ↪ flag_sersic})
table
```

[14]:

	object_id	gini	M_20	flag_sersic
0	173	0.393321	-0.742974	1
1	1094	0.741822	-0.472583	1
2	1156	-99.000000	-99.000000	1
3	1326	-99.000000	-99.000000	1
4	1348	0.580671	-0.978229	1
..	…	…	…	…
127	24771	0.420904	-1.044237	0
128	24894	0.450245	-0.745930	1
129	24944	0.537971	-1.733450	1
130	24968	-99.000000	-99.000000	1
131	24996	0.543428	-1.637970	0

[132 rows x 4 columns]

8. The merger population can be differentiated by the $Gini - M_{20}$ values given in Rodriguez-Gomez et al.(2019). The `source_morphology` automatically computes this as a statistic given by `morph.gini_m20_merger`. The reason $Gini$ and M_{20} values are stored as separate arrays is so that we can generate a plot from it. The code below is used to draw the line between mergers and non mergers.

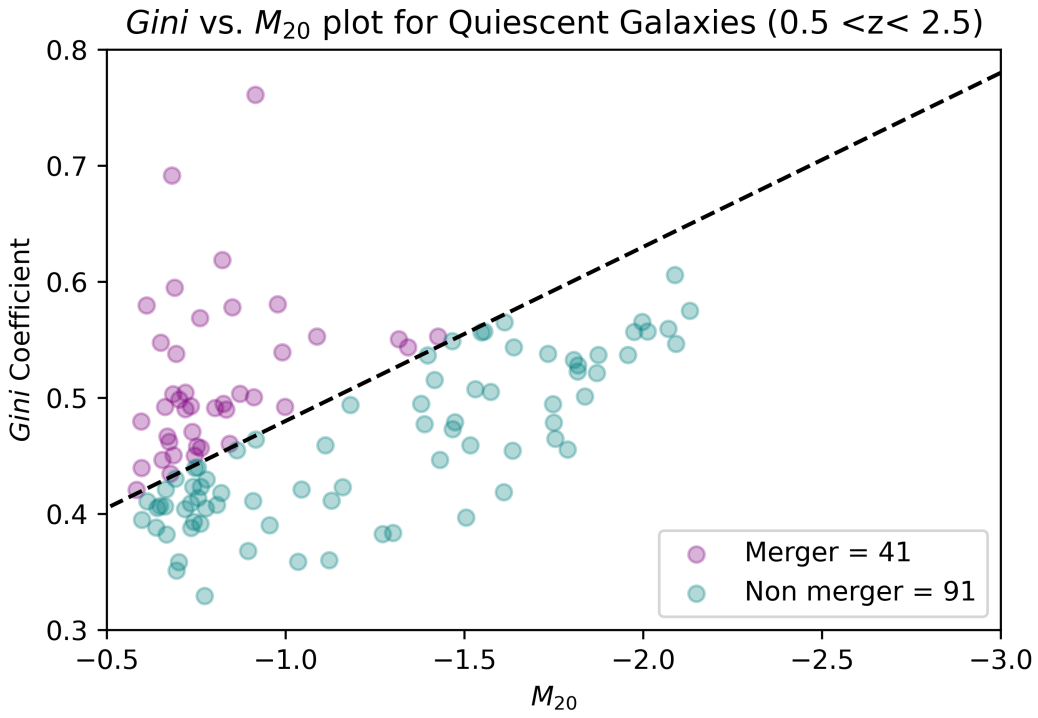
[15]:
```
# merger vs non-merger cutoff
merger_sel = (table['gini'] > (-0.15*table['M_20'] + 0.33)) #merger
n_merger_sel = (table['gini'] <= (-0.15*table['M_20'] + 0.33)) #non merger
```

9. The $Gini$ vs M_{20} plot! The plot shows that the quiescent galaxies with merger signatures are on the top left half of the plot. Now that we have identified the mergers by visualizing it in a plot, it is time to use their object IDs to perform further morphological analysis by plotting the $Gini$ vs M_{20} values obtained from `statmorph`.

[16]:
```
cutoff = np.arange(-3,-0.5,0.01)*(-0.15)+0.33
plt.figure(dpi=500,facecolor='white')
plt.scatter(table['M_20'][merger_sel],table['gini'][merger_sel],color='purple',
            alpha=0.3,label='Merger = '+str(len(table['M_20'][merger_sel])))
plt.
 ↪scatter(table['M_20'][n_merger_sel],table['gini'][n_merger_sel],color='teal',
            alpha=0.3,label='Non merger =␣
 ↪'+str(len(table['M_20'][n_merger_sel])))
plt.plot(np.arange(-3,-0.5,0.01),cutoff,'k--')
plt.title('$Gini$ vs. $M_{20}$ plot for Quiescent Galaxies (0.5 <z< 2.5)')
plt.xlabel('$M_{20}$')
plt.ylabel('$Gini$ Coefficient')
```

```
plt.xlim([-0.5,-3])
plt.ylim([0.3,0.8])
plt.legend(loc='lower right')
```

[16]: <matplotlib.legend.Legend at 0x7f7778a30250>



10. An example quiescent merging galaxy from the object list is shown below. The code is entirely adapted form the statmorph tutorial (under '**Examining Sersic profile**').The Original image shows 3 galaxies merging. After subtraction of the fit for the target galaxy, we can see the some tidal effects in the residual. We can plot the same for other objects in the list to look at the morphology of merging galaxies.

[17]:
```
#Print out object IDs of merger candidates
table['object_id'][merger_sel]
```

[17]:
```
1        1094
4        1348
6        1822
7        1824
8        1854
9        2040
11       2333
14       3263
26       5467
```

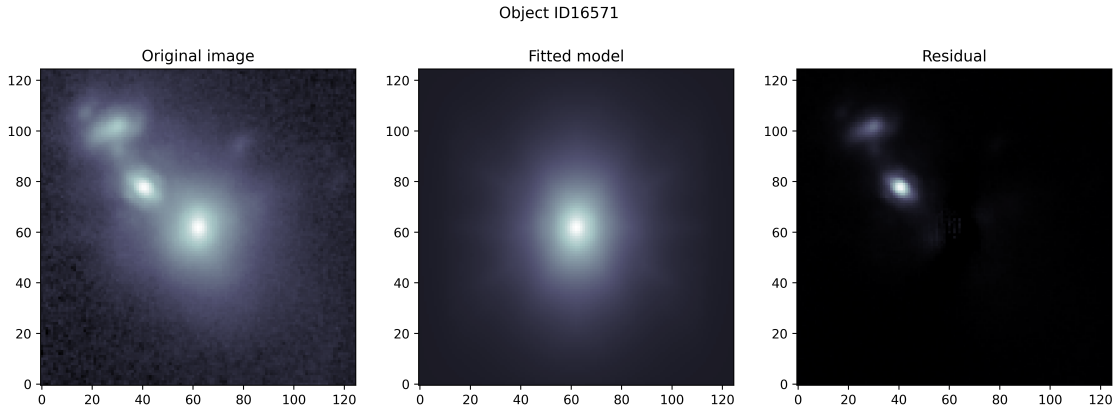
```
27       5620
32       6250
43       8384
44       8498
48       9242
49       9243
52       9469
56      10405
57      10522
58      10746
61      12687
62      13084
65      13803
73      15630
77      16572
79      17112
81      17191
84      17858
87      18851
89      19038
92      20109
93      20245
97      20953
104     21599
105     22003
109     22729
111     22968
112     23056
116     23602
122     24379
124     24476
128     24894
Name: object_id, dtype: int64
```

[18]:
```
#Source_morphology on a sample merger candidate
object_id = 16571
size = 5 #set the size of the cutout
obj = ceers[ceers['id']==object_id]
pos = SkyCoord(obj['ra'],obj['dec'])
image = Cutout2D(sci,pos,size*u.arcsec,wcs=w).data
objseg = Cutout2D(seg.copy(),pos,size*u.arcsec,wcs=w).data
wht = Cutout2D(wht_map.copy(),pos,size*u.arcsec,wcs=w).data
segcopy = Cutout2D(seg.copy(),pos,size*u.arcsec,wcs=w).data
#mask = (segcopy!=object_id) | (segcopy==0) # True only for sky pixels or the␣
 ↪main object, so other objects are masked.
mask = (segcopy!=object_id) & (segcopy>0)
#revmask = (segcopy!=object_id) & (segcopy>0)
```

```
npmask = np.array(mask, dtype='bool8')

source_morphs = statmorph.source_morphology(image, objseg,label=object_id,␣
↪weightmap=wht, psf=psf)
morph = source_morphs[0]

#create a model using sersic values
ny, nx = image.shape
y, x = np.mgrid[0:ny, 0:nx]
fitted_model = statmorph.ConvolvedSersic2D(
    amplitude = morph.sersic_amplitude,
    r_eff = morph.sersic_rhalf,
    n=morph.sersic_n,
    x_0=morph.sersic_xc,
    y_0=morph.sersic_yc,
    ellip=morph.sersic_ellip,
    theta=morph.sersic_theta)
fitted_model.set_psf(psf)  # required when using ConvolvedSersic2D
image_model = fitted_model(x, y)
```

[21]:
```
#Plot results and fit!
bg_noise = 0 #if background flux is known enter it here
fig = plt.figure(figsize=(15,5),dpi=500,facecolor='white')
plt.suptitle('Object ID'+ str(object_id))
ax = fig.add_subplot(131)
ax.imshow(image, cmap='bone', origin='lower',
            norm=simple_norm(image, stretch='log'))
ax.set_title('Original image')

ax = fig.add_subplot(132)
residual = (image - image_model)
ax.imshow(image_model + bg_noise, cmap='bone', origin='lower',
            norm=simple_norm(image, stretch='log'))
ax.set_title('Fitted model')

ax = fig.add_subplot(133)
ax.imshow(residual, cmap='bone', origin='lower',
          norm=simple_norm(image, stretch='asinh')) # using stretch=asinh␣
↪reduces prominance of background
ax.set_title('Residual')
```

[21]:
```
Text(0.5, 1.0, 'Residual')
```



5 Conclusion

We see that `statmorph` is a very useful tool to identify mergers. This tutorial presents a more streamlined way of identifying galaxy candidates that have undergone mergers by plotting the *Gini* vs. M_{20} plot and then picking out the merger population from the plot. This method helps us to visualize the data on mergers from catalogs such as CEERS for a desired sub-group of galaxies. For further investigation, we can try getting the *Gini* vs. M_{20} plots for star-forming galaxies or galaxies at higher redshifts to see how the merger ratios vary. The investigation done in this tutorial shows that about 41/132 (~30%) of massive, quiescent galaxies at redshifts $0.5 < z < 2.5$ are mergers.

6 References

Bruce, V. A., Dunlop, J. S., Cirasuolo, M., McLure, R. J., Targett, T. A., Bell, E. F., Croton, D. J., Dekel, A., Faber, S. M., Ferguson, H. C., Grogin, N. A., Kocevski, D. D., Koekemoer, A. M., Koo, D. C., Lai, K., Lotz, J. M., McGrath, E. J., Newman, J. A., & van der Wel, A. (2012). The morphologies of massive galaxies at 1 <z< 3 in the Candels-UDS field: Compact bulges, and the rise and fall of massive discs. Monthly Notices of the Royal Astronomical Society, 427(2), 1666–1701. https://doi.org/10.1111/j.1365-2966.2012.22087.x

Ellison, S. L., Wilkinson, S., Woo, J., Leung, H.-H., Wild, V., Bickley, R. W., Patton, D. R., Quai, S., & Gwyn, S. (2022). Galaxy mergers can rapidly shut down star formation. Monthly Notices of the Royal Astronomical Society: Letters, 517(1). https://doi.org/10.1093/mnrasl/slac109

Fang, J. J., Faber, S. M., Koo, D. C., Rodríguez-Puebla, A., Guo, Y., Barro, G., Behroozi, P., Brammer, G., Chen, Z., Dekel, A., Ferguson, H. C., Gawiser, E., Giavalisco, M., Kartaltepe, J., Kocevski, D. D., Koekemoer, A. M., McGrath, E. J., McIntosh, D., Newman, J. A., … Wuyts, S. (2018). Demographics of star-forming galaxies since z 2.5. i. the uvj diagram in Candels. The Astrophysical Journal, 858(2), 100. https://doi.org/10.3847/1538-4357/aabcba

Rodriguez-Gomez, V., Snyder, G. F., Lotz, J. M., Nelson, D., Pillepich, A., Springel, V., Genel, S., Weinberger, R., Tacchella, S., Pakmor, R., Torrey, P., Marinacci, F., Vogelsberger, M., Hernquist, L., & Thilker, D. A. (2018). The optical morphologies of galaxies in the ILLUSTRISTNG Simu-

lation: A comparison to Pan-STARRS observations. Monthly Notices of the Royal Astronomical Society, 483(3), 4140–4159. https://doi.org/10.1093/mnras/sty3345

Whitaker, K. E., Kriek, M., van Dokkum, P. G., Bezanson, R., Brammer, G., Franx, M., & Labbé, I. (2012). A large population of massive compact post-starburst galaxies atz> 1: Implications for the size evolution and quenching mechanism of quiescent galaxies. The Astrophysical Journal, 745(2), 179. https://doi.org/10.1088/0004-637x/745/2/179

[ ]: